# Prise de décision contextuelle en bande organisée : Quand les bandits font un brainstorming

Robin Allesiardo, Raphael Féraud, Djallel Bouneffouf

# Prise de décision contextuelle en bande organisée : Quand les bandits font un brainstorming

Robin Allesiardo[1,2], Raphaël Féraud[1], et Djallel Bouneffouf[1]

[1]Orange Labs
[2]TAO - INRIA, CNRS, Université Paris-Sud 11

13 juin 2014

## Résumé

Dans cet article, nous proposons un nouvel algorithme de bandits contextuels, NeuralBandit, ne faisant aucune hypothèse de stationnarité sur les contextes et les récompenses. L'algorithme proposé utilise plusieurs perceptrons multicouches, chacun apprenant la probabilité qu'une action, étant donné le contexte, entraine une récompense. Afin de régler en ligne les paramètres de ces perceptrons multicouches, et notamment les architectures, nous proposons d'utiliser une approche multi-experts. Des tests sur des jeux de données synthétiques et réels montrent l'apport de l'algorithme NeuralBandit par rapport à l'état de l'art.

**Mots-clef** : Bandits-manchots contextuels, apprentissage par renforcement, réseaux de neurones, non stationnarité

## 1 Introduction

Dans la plupart des problèmes de décision rencontrés sur le web (publicité en ligne, optimisation marketing), les algorithmes doivent être capables d'apprendre à partir d'une information partielle, sans connaitre quelle était la meilleure décision. Par exemple, dans le cas de la publicité en ligne, un visiteur arrive sur une page ; une requête contenant le contexte (page demandée, cookies de l'utilisateur, profil client, etc.) est envoyée à un serveur d'optimisation publicitaire ; le serveur renvoie une suggestion ; la publicité suggérée est présentée au visiteur ; le serveur reçoit une récompense si le visiteur clique sur le lien. Rien n'indique au serveur de publicité quelle était la meilleure décision (celle correspondant à la probabilité de clic maximale).

De plus, dans un cadre applicatif, la stationnarité des données peut être remise en cause. En effet, la distribution des récompenses en fonction des contextes et des actions peut changer au cours du temps par exemple à cause d'un changement de comportement des cookies vis à vis d'une publicité, de la présentation d'une publicité concurrente sur un emplacement proche, du changement de contenu de la page... Les algorithmes ont donc un fort intérêt à être robustes à la non-stationnarité afin de pouvoir être utilisés sur une longue période. Les solutions proposées dans l'état de l'art comportent cependant différents points faibles. Elles sont souvent linéaires, ne prennent pas en compte la non-stationnarité, souffrent d'explosion combinatoire ou bien comportent des 'oracles'.

Nous allons essayer d'apporter des débuts de réponses aux questions suivantes :

- Quelle pourrait-être l'alternative aux modèles linéaires ?
- Comment appréhender la non-stationnarité ?
- Si ce modèle comporte des paramètres, comment les choisir en ligne ?

## 2 Travaux précédents

Le problème des bandits manchots consiste à choisir quel bras (ou action) jouer, parmi ceux de $K$ machines à sous supposées non identiques, de manière à maximiser les gains cumulés. Résoudre ce problème revient à trouver le bon compromis entre l'exploration (jouer chaque bras pour juger de sa performance) et l'exploitation (jouer le bras qui semble donner les meilleures récompenses). Ce problème assez ancien a été largement étudié. Des solutions optimales ont été proposées pour des récompenses tirées suivant des distributions de moyennes inconnues [LR85, ACBF02], ou tirées sui-

vant une hypothèse bayésienne [Tho33, KKM12], ou pour une séquence de récompenses choisies à l'avance par un adversaire [ACB98, ACBFS02].

Pour satisfaire des contraintes pratiques (apparition d'une nouvelle publicité après le début de l'apprentissage, un nombre d'affichages contractuel à respecter) des variantes du problème initial ont été introduites [KNMS08, CKRU08, FU12, FU13]. Cependant ces approches ne prennent pas en compte le contexte alors que la performance des bras pourrait être corrélée avec celui-ci.

Une solution naïve au problème des bandits contextuels consiste à allouer autant de problèmes de bandits qu'il y a de contextes possibles, c'est à dire le produit cartésien des ensembles de valeurs des variables de contexte. L'utilisation d'une structure d'arbre, comme dans X-armed bandits [BMSS08] ou dans une variante d'UCT [KS06] réduisant le nombre de variables avec une méthode similaire à FUSE [GS10], pour modéliser les différentes combinaisons de variables contextuelles, permet de faciliter l'exploration et l'exploitation d'un grand nombre de bras en permettant aux contextes proches de mutualiser leurs solutions. L'aspect fortement combinatoire de ces approches ne permet pas l'allocation mémoire des problèmes contenant trop de variables. Dans [SAL+11], les contextes sont modélisés par un ensemble d'états qui sont chacun associés à un problème de bandits. Sans connaissance à priori de l'espace des contextes, il est nécessaire d'utiliser un contexte par état, ce qui est équivalent à l'approche naïve. Dudík et al [DHK+11] proposent quant à eux un algorithme d'élimination de politiques qui possède les mêmes limites que l'algorithme précédent : les performances dépendent de la présence d'une bonne politique dans l'ensemble des politiques testées. L'algorithme Epoch-Greedy [LZ07] alterne quant à lui les cycles d'exploration et d'exploitation. Lors des cycles d'exploration, le bras choisi est tiré aléatoirement et l'ensemble des contextes-bras-récompenses obtenus jusqu'à présent sont utilisés pour apprendre le classifieur qui sera utilisé lors du cycle d'exploitation suivant. La nature de ce classifieur reste cependant à définir pour une utilisation concrète. Banditron [KSST08] utilise un perceptron [Ros58] par action pour reconnaitre les contextes récompensés mais est limité aux cas linéairement séparables. Dans LINUCB [LCLS10, CLRS11] et dans le Thompson Sampling Contextuel(CTS) [AG12] les auteurs modélisent la dépendance entre les actions et les récompenses en utilisant des prédicteurs linéaires. Par ailleurs, les fondations théoriques sur lesquelles reposent ces algorithmes supposent que les données et les récompenses sont tirées selon une distribution stationnaire, ce qui limite leur usage pratique.

L'algorithme EXP4 [ACBFS02] permet de choisir le meilleur bras en connaissant les avis de $N$ experts, représentés par des vecteurs contenant la probabilité de chaque bras. La recherche du lien entre les récompenses des bras et le contexte est ainsi déléguée aux experts. Contrairement aux algorithmes cités précédemment, ici, les récompenses sont supposées choisies par un adversaire, cet algorithme peut donc s'appliquer sur des données non stationnaires.

Dans un premier temps nous poserons de manière formelle le problème des bandits contextuels et présenterons un premier algorithme : NeuralBandit1. Celui-ci s'inspire de Banditron [KSST08] et estime les probabilités de récompenses à l'aide de réseaux de neurones afin de s'affranchir de l'hypothèse de séparation linéaire des données. En effet, les réseaux de neurones sont des approximateurs universels [HSW89]. Leur utilisation en apprentissage par renforcement [Tes02, MKS+13] a montré qu'ils pouvaient estimer de manière précise les probabilités de récompenses dans le cadre de problèmes réels et complexes. De plus les algorithmes d'apprentissage par descente de gradient font parti de la famille des algorithmes d'apprentissage stochastique qui permettent d'obtenir de bonnes performances en termes de convergence vers le point de meilleure généralisation [BL05] et ont l'avantage d'apprendre en ligne. La descente de gradient, en cherchant à atteindre un minimum local, peut permettre de résister à la non stationnarité. Celle-ci se traduit par un changement du paysage de la fonction de coût au cours du temps. Si le paysage de la fonction de coût change avec une vitesse raisonnable, l'algorithme poursuivra la descente vers un nouveau minimum local.

Le principal problème soulevé par l'utilisation des perceptrons multicouches pour l'apprentissage en ligne reste le réglage des différents paramètres comme le nombre de neurones cachés, la valeur du pas d'apprentissage, la graine d'initialisation des poids... C'est pourquoi, nous proposons ensuite deux versions avancées de l'algorithme permettant de régler ces paramètres à l'aide d'algorithmes de bandits non-stochastiques qui cherchent, parmi plusieurs modèles initialisés avec des paramètres différents, celui qui a les meilleures performances. Nous terminerons en confrontant ces différentes approches à l'état de l'art sur des flux de données stationnaires puis non stationnaires.

# 3 Bandits contextuels

## 3.1 Définitions

**Bandits contextuels** Soit une séquence $((x_1, \mathbf{y}_1), ..., (x_T, \mathbf{y}_T))$ avec $x_t \in X$ un contexte, $k \in [K] = \{1, ..., K\}$ un des $K$ bras pouvant être choisis et $\mathbf{y}_t \in Y$ un vecteur de récompense $(y_{t,1}, ..., y_{t,K})$ avec $y_{t,k} \in [0, 1]$ la récompense pour le bras $k$. La séquence peut-être être tirée selon un processus stochastique ou définie à l'avance par un adversaire. Le problème est répété sur $T$ tours : à chaque tour $t < T$ le contexte $x_t$ est annoncé. Le joueur, qui cherche à maximiser la somme de ses récompenses, choisit un bras $k_t$. La récompense $y_{t,k}$ du bras choisi par le joueur, et uniquement celle-ci, est dévoilée.

**Regret cumulé** Soit $H : X \to [K]$ un ensemble d'hypothèses, $h_t \in H$ une hypothèse trouvée par l'algorithme $A$ au tour $t$ et $h_t^* = \underset{h_t \in H}{\operatorname{argmax}}\, y_{t,h_t(x_t)}$ l'hypothèse optimale au même tour. Le regret instantané est :

$$R_t(A) = y_{t,h_t^*(x_t)} - y_{t,h_t(x_t)}$$

Le regret cumulé est :

$$R(A) = \sum_{t=1}^{T} R_t(A)$$

Le but d'un algorithme de bandits contextuels est de minimiser le regret cumulé.

## 3.2 NeuralBandit1

Nous présentons maintenant NeuralBandit1 (algorithme 1), un algorithme de bandits contextuels inspiré du Banditron [KSST08]. Une approche "un contre tous" est utilisée. Chaque action $k$ est associée à un réseau de neurones à une couche cachée qui cherche à apprendre la probabilité de récompense d'une action sachant son contexte. C'est l'estimation de cette probabilité qui sera utilisée comme score pour chaque action.

Soit $K$ le nombre d'actions, $C$ le nombre de neurones composant les couches cachées des réseaux et $\mathbf{N}_t^k : X \to Y$ la fonction associant un contexte $x_t$ à la sortie du réseau de neurones correspondant à l'action $k$ au tour $t$. Le nombre de connexions pour chaque réseau est noté $N$ avec $N = dim(X)C + C$. Pour faciliter les notations, nous plaçons l'ensemble des connexions dans la matrice $W_t$ de taille $K \times N$, ainsi chaque ligne de la matrice $W_t$ contient les poids d'un réseau. $\Delta_t$ est la matrice de taille $K \times N$ contenant les modifications de chaque poids entre le tour $t$ et le tour $t+1$. L'équation de mise à jour des poids est

$$W_{t+1} = W_t + \Delta_t$$

L'algorithme de rétro-propagation du gradient [RHW86] permet de calculer le gradient de l'erreur pour chaque neurone, de la dernière couche vers la première en minimisant une fonction de coût (nous utilisons la fonction d'erreur quadratique et la fonction d'activation sigmoïde).

Soit $\lambda$ le pas d'apprentissage, $\hat{x}_t^{n,k}$ la valeur de l'entrée associée à la connexion $n$ du réseau $k$ et $\delta_t^{n,k}$ le gradient de la fonction d'erreur au tour $t$ pour le neurone possédant la connexion $n$ du réseau $k$. $\Delta_t^{n,k}$ est la valeur correspondant aux coordonnées $(n, k)$ de la matrice $\Delta_t$. Lorsque la récompense d'un bras est connue, on peut calculer :

$$\Delta_t^{n,k} = \lambda \hat{x}_t^{n,k} \delta_t^{n,k}$$

Étant dans un cas à information partielle, seule la récompense obtenue en jouant le bras $k_t$ est disponible. Pour apprendre la meilleure action à jouer, une première approche consisterait à effectuer une première phase d'exploration où chaque action serait jouée le même nombre de fois. Ainsi, l'estimateur obtenu ne serait pas biaisé sur les actions les plus jouées. Cependant, une telle approche aurait des performances catastrophiques en cas de non-stationnarité des données. Notre approche consiste à garder un facteur d'exploration $\gamma$ constant au cours du temps, permettant de poursuivre la mise à jour des modèles en cas de non-stationnarité des données. La probabilité de jouer le bras $k$ au tour $t$ en sachant que $\hat{k}_t$ est le bras possédant la plus haute prédiction de récompense est :

$$\mathbf{P}_t(k) = (1 - \gamma)\mathbf{1}[k = \hat{k}_t] + \frac{\gamma}{K}$$

Pour tenir compte de cette probabilité de tirage des actions, nous proposons une modification de la règle d'apprentissage :

$$\tilde{\Delta}_t^{n,k} = \frac{\lambda \hat{x}_t^{n,k} \delta_t^{n,k} \mathbf{1}[\hat{k}_t = k]}{\mathbf{P}_t(k)}$$

**Proposition 1.** *L'espérance de $\tilde{\Delta}_t^{n,k}$ est égale à $\Delta_t^{n,k}$.*

*Démonstration.*

$$\begin{aligned} \mathbf{E}[\tilde{\Delta}_t^{n,k}] &= \sum_{k=1}^{K} \mathbf{P}_t(k) \left(\frac{\lambda \hat{x}_t^{n,k} \delta_t^{n,k} \mathbf{1}[\hat{k}_t = k]}{\mathbf{P}_t(k)}\right) \\ &= \lambda \hat{x}_t^{n,k} \delta_t^{n,k} \\ &= \Delta_t^{n,k} \end{aligned}$$

□

La nouvelle équation de mise à jour des poids est :

$$W_{t+1} = W_t + \tilde{\Delta}_t \quad (1)$$

L'algorithme proposé, NeuralBandit1, peut s'adapter à des non-stationnarités en continuant à apprendre au cours du temps, tout en obtenant en moyenne le même résultat que si nous avions appris les poids dans une première phase d'exploration, où chaque action serait jouée le même nombre de fois.

Procédons à un récapitulatif du fonctionnement de notre algorithme (voir algorithme 1) :

- Chaque action $k$ est associée à un réseau.
- Lorsqu'un contexte $x_t$ est présenté à l'algorithme le score $\mathbf{N}_t^k(x_t)$ de chaque action est calculé.
- Une exploration des actions peut avoir lieu avec une probabilité $\gamma$.
- S'il y a exploration, l'action à jouer est tirée uniformément dans l'ensemble des actions, sinon l'action obtenant le plus haut score est jouée.
- Après avoir reçu la récompense, les poids du réseau correspondant à l'action jouée sont mis à jour.

**Algorithm 1:** NeuralBandit1

**Data**: $\gamma \in [0, 0.5]$ et $\lambda \in ]0, 1]$

**begin**

Initialisation de $W_1 \in ]-0.5, 0.5[^{N \times K}$

**for** $t = 1, 2, ..., T$ **do**

Le contexte $x_t$ est dévoilé

$\hat{k}_t = \arg\max_{k \in [K]} \mathbf{N}_t^k(x_t)$

$\forall k \in [K]$ on a $\mathbf{P}_t(k) = (1-\gamma)\mathbf{1}[k = \hat{k}_t] + \frac{\gamma}{K}$

On tire $\tilde{k}_t$ suivant $\mathbf{P}_t$

On prédit $\tilde{k}_t$ et on reçoit la récompense $y_{t,\tilde{k}_t}$

On définit $\tilde{\Delta}_t$ tel que $\tilde{\Delta}_t^{n,k} = \frac{\lambda \hat{x}_t^{n,k} \delta_t^{n,k} \mathbf{1}[k_t = k]}{\mathbf{P}_t(k)}$

$W_{t+1} = W_t + \tilde{\Delta}_t$

# 4 Sélection de modèles

## 4.1 Avant propos

Les performances des réseaux de neurones dépendent de plusieurs paramètres, comme le pas d'apprentissage, le nombre de couches cachées, leur taille ou les valeurs d'initialisation des poids. Nous appelons modèle, le prédicteur initialisé avec ces paramètres. Lors des problèmes d'apprentissage hors-ligne, la sélection de modèle est réalisée en utilisant un ensemble de validation. Dans le cas de l'apprentissage en ligne, les modèles sont entrainés en parallèle sur le flux de données. Pour sélectionner le meilleur modèle en ligne, nous proposons d'utiliser l'algorithme de bandits non-stochastiques Exp3 [ACB98, ACBFS02].

**Exp3** Soit le paramètre d'exploration $\gamma_{\text{model}} \in [0, 1]$ et le vecteur de poids $w_t$ avec $w_t^k = 1$ et $m \in \{1, ..., M\}$. La probabilité de choisir le modèle $m$ au tour t est :

$$\mathbf{P}_{\text{model}t}(m) = (1 - \gamma_{\text{model}}) \frac{w_t^m}{\sum_{i=1}^{M} w_t^i} + \frac{\gamma_{\text{model}}}{M} \quad (2)$$

Soit $y_{t,\hat{m}}$ la récompense reçue après avoir joué le modèle $\hat{m}$. L'équation de mise à jour des poids est :

$$w_{t+1}^i = w_t^i \exp\left(\frac{\gamma_{\text{model}} \mathbf{1}[i = \hat{m}] y_{t,\hat{m}}}{\mathbf{P}_{\text{model}t}(i) M}\right) \quad (3)$$

## 4.2 NeuralBandit2

L'idée est d'instancier un algorithme de bandits non-stochastiques qui cherchera, parmi les modèles générés à partir des paramètres proposés par l'utilisateur, le plus performant. L'algorithme prend en paramètre une liste contenant $M$ paramétrages et un paramètre d'exploration de modèle $\gamma_{\text{model}}$. Pour chaque élément de la liste, une instance de NeuralBandit1 est initialisée et représente un bras d'un algorithme EXP3. C'est celui-ci qui définira quel modèle choisira le bras à jouer à chaque tour. Après avoir obtenu la récompense, les réseaux de neurones correspondant à l'action jouée seront mis à jour pour chaque modèle, ainsi que l'EXP3.

## 4.3 NeuralBandit3

L'algorithme NeuralBandit2 cherche le modèle, parmi ceux proposés, permettant d'estimer la probabilité de récompense de l'ensemble des actions de manière optimale. On peut cependant supposer que pour certaines actions les meilleurs modèles pourraient être différents. L'algorithme NeuralBandit3 permet aux actions d'être associées à des modèles différents. Pour chaque action, un problème de bandits non-stochastiques différent est utilisé pour chercher le meilleur modèle. Une probabilité d'exploration $\gamma$ est ensuite introduite dans le choix du bras à jouer.

**Algorithm 2:** NeuralBandit2

**Data**: $\gamma_{\text{model}} \in [0, 0.5]$ et un ensemble de $M$ paramétrage de modèles

**begin**

Initialisation des $M$ NeuralBandit1

Initialisation du vecteur de poids $w_0$ d'EXP3 avec $\forall m \in [M]\ w_0^m = 1$

**for** $t = 1, 2, ..., T$ **do**

Le contexte $x_t$ est dévoilé

$m_t$ est tiré selon $\mathbf{P}_{\text{model}t}$ (2)

Le modèle $m_t$ choisit l'action $\tilde{k}_t$

$\tilde{k}_t$ est prédit et la récompense $y_{t,\tilde{k}_t}$ est reçue

Mise à jour des réseaux correspondant à l'action $\tilde{k}_t$ pour chaque modèle avec (1)

Mise à jour des vecteurs de poids des EXP3 avec (3)

**Algorithm 3:** NeuralBandit3

**Data**: $\gamma \in [0, 0.5]$, $\gamma_{\text{model}} \in [0, 0.5]$ et un ensemble de $M$ modèles

**begin**

Initialisation de $K$ réseaux de neurones pour chaque modèle $m$

Initialisation de $K$ EXP3

**for** $t = 1, 2, ..., T$ **do**

Le contexte $x_t$ est dévoilé

**for** $k = 1, 2, ..., K$ **do**

$m_t^k$ est tiré suivant $\mathbf{P}_{\text{model}_t^k}$ (2)

L'action $k$ reçoit le score $s_t^k = \mathbf{N}_t^{m_t^k,k}(x_t)$

$\hat{k}_t = \arg\max_{k \in [K]} s_t^k$

$\forall k \in [K]$ on a $\mathbf{P}_t(k) = (1-\gamma)\mathbf{1}[k = \hat{k}_t] + \frac{\gamma}{K}$

$\tilde{k}_t$ est tiré suivant $\mathbf{P}_t$

$\tilde{k}_t$ est prédit et la récompense $y_{t,\tilde{k}_t}$ est reçue

Mise à jour des réseaux correspondant à l'action $\tilde{k}_t$ pour chaque modèle avec (1)

Mise à jour des vecteurs de poids de chaque l'EXP3 avec (3)

NeuralBandit3 possède une plus grande capacité d'expression que NeuralBandit2 en permettant d'associer des modèles différents à chaque action. NeuralBandit2 devrait cependant trouver un ensemble de modèles plus rapidement car il ne possède qu'une seule instance d'EXP3 à mettre à jour.

# 5 Expérimentations

Les courbes présentées sont obtenues en moyennant 20 exécutions de chaque algorithme avec $\gamma = 0.005$, $\gamma_{model} = 0.1$.

## 5.1 Expérimentations sur des données stationnaires

Dans un premier temps nous allons vérifier la validité des algorithmes de sélection de modèles sur un problème synthétique. Les contextes contiennent 20 variables binaires, sont tirés de manière uniforme et sont associés à 10 actions. La récompense moyenne de chaque action sachant le contexte est tirée selon une distribution de Pareto de paramètres $x_m = 1$ et $k = 5$ pour chaque combinaison de valeurs des trois premières variables. Les 17 autres variables ne contiennent pas d'information.

La figure 1 montre le regret cumulé de plusieurs NeuralBandit1 contenant des réseaux ayant des couches cachées de tailles $C$ différentes (1, 5, 25, 50 et 100) et $\lambda = 10^{-1}$. Les deux algorithmes de sélection de modèles obtiennent en moyenne un regret plus faible que le meilleur modèle à cause de la variance des performances lors des différentes exécutions. Bien que le modèle ayant une couche cachée de taille $C = 5$ soit le plus performant, le regret de $C = 25$ est très proche. Suivant les exécutions, le choix de l'algorithme alterne entre ces deux modèles. Pour illustrer cela, l'algorithme NeuralBandit3 est exécuté sur 10 groupes de réseaux de taille $C = 5$. On remarque que le regret cumulé moyen des meilleurs réseaux à chaque exécution est plus faible que celui de la moyenne des réseaux de même taille sur l'ensemble des exécutions car les modèles ayant le minimum local le plus favorable sont sélectionnés.

La seconde expérimentation (figure 2) est effectuée sur le jeu de données Forest Cover Type provenant de l'UCI Machine Learning Repository. Les variables continues ont été discrétisés en 5 variables binaires avec un à priori de fréquences égales par intervalles. Les contextes contiennent 94 variables binaires et sont

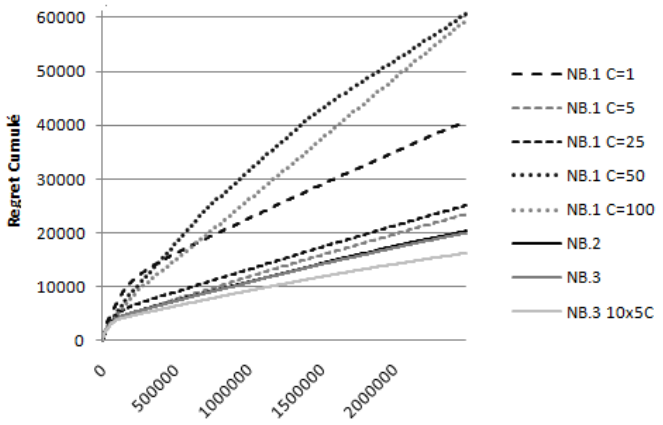


FIGURE 1 – Le regret cumulé des différentes versions de NeuralBandit. Les algorithmes de sélection de modèles obtiennent le plus petit regret.

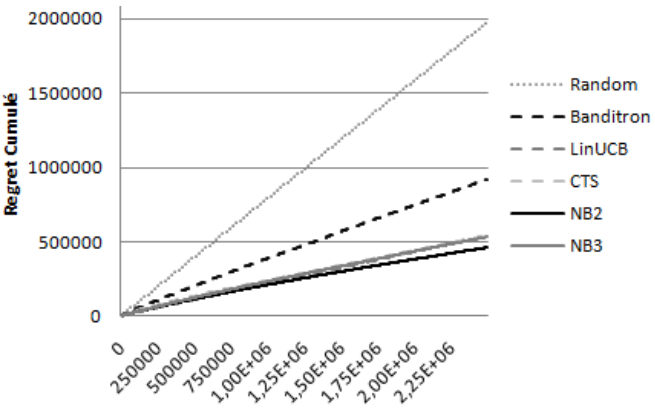


FIGURE 2 – Le regret cumulé de différents algorithmes de bandits contextuels au cours du temps sur le jeu de données Forest Cover Type. NeuralBandit2 obtient le plus petit regret.

associés à 7 classes différentes. Chaque action correspond à une classe. Lorsque la classe choisie par l'algorithme est correcte, la récompense est de 1, sinon elle est de 0. Le jeu de données contient 581000 instances et est mélangé. On boucle sur celui-ci afin de simuler un flux et d'atteindre 5 millions d'itérations. La politique optimale utilisée pour calculer le regret est un arbre de décision ayant appris par coeur le jeu de données et atteignant 93% de bonnes classifications. Les algorithmes de sélection de modèles sont utilisés avec les mêmes tailles que précédemment et plusieurs $\lambda$ différents $(10^{-2}, 10^{-1}, 1)$. La combinatoire de ce jeu de données est trop élevée pour UCB et UCT qui ne peuvent pas être instanciés ($\approx 10^{27}$ contextes différents possibles). Banditron atteint un regret cumulé élevé avec un taux de classification de 57% sur les 100 000 dernières prédictions) et est moins performant que les autres algorithmes contextuels sur ce jeu de données. Le regret cumulé ainsi que le taux de classification final de LinUCB (72%), CTS (73%) et NeuralBandit3 (73%) sont très proches, les trois courbes étant confondues. La figure 3 décrit l'augmentation moyenne du regret à chaque itération. LinUCB converge très vite vers sa solution finale tandis que les algorithmes de sélection d'architecture ainsi que CTS sont plus lents mais trouvent une meilleure solution.

## 5.2 Expérimentations sur des données non-stationnaires

Au cours de cette expérimentation nous avons introduit un changement de stationnarité brusque sur les classes associées aux exemples de Forest Cover

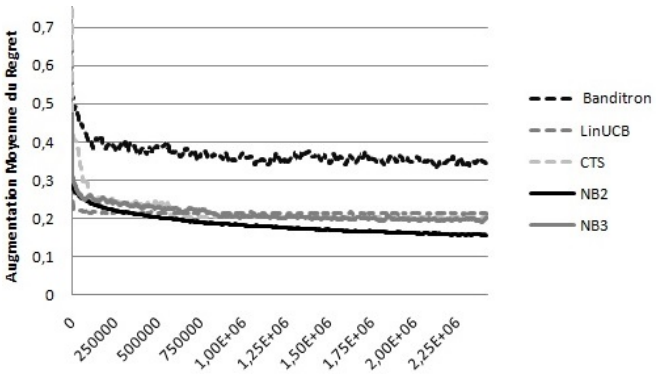


FIGURE 3 – L'augmentation moyenne du regret au cours du temps sur le jeu de données Forest Cover Type.

Type. Toutes 500 000 itérations, les classes sont interchangées selon un cycle circulaire ($1 \rightarrow 2$, $2 \rightarrow 3$ ,...,$7 \rightarrow 1$) afin de simuler une dérive de concept. Les résultats sont présentés en figure 4.

À chaque changement de classe, les pentes des courbes de regret de NeuralBandit2, NeuralBandit3 et Banditron augmentent avant de se stabiliser. Banditron converge à nouveau environ 50 000 itérations plus tard alors que les deux algorithmes de sélection de modèles nécessitent entre 75 000 et 350 000 itérations dans le pire des cas (lors du premier changement de stationnarité). Malgré le fait que NeuralBandit2 et NeuralBandit3 soient des modèles plus complexes, ils outrepassent les performances de Banditron dans un contexte non stationnaire sur ce jeu de données. LinUCB et CTS ne résistent pas à la première dérive

et voient leurs performances se dégrader fortement. La figure 5 décrit l'augmentation moyenne du regret à chaque itération. NeuralBandit3 converge plus rapidement que les autres algorithmes testés en cas de concept drift (Figure 5). En effet, la diversité des modèles maintenus dans NeuralBandit3 favorise une plus grande diversité des paysages de la fonction d'erreur et donc multiplie les chances de trouver pour chaque action un point de départ favorable à la descente de grandient lors du concept drift.

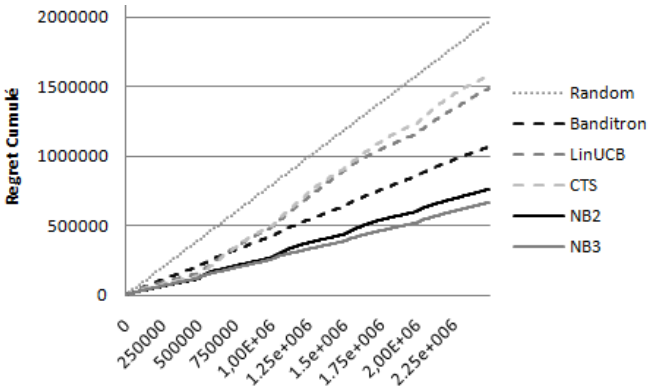


FIGURE 4 – Le regret cumulé de différents algorithmes de bandit contextuels au cour du temps sur le jeu de données Forest Cover Type soumit à une dérive de concept toutes les 500 000 itérations

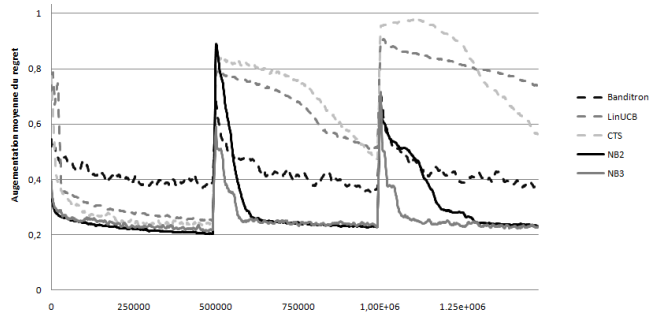


FIGURE 5 – L'augmentation moyenne du regret au cours du temps sur le jeu de données Forest Cover Type soumit à une dérive de concept toutes les 500.000 itérations.

# 6 Conclusion

Nous avons présenté un nouvel algorithme de bandits contextuels NeuralBandit1 et l'avons confronté à des données synthétiques et réelles. Celui-ci obtient un plus petit regret cumulé que d'autres algorithmes de l'état de l'art. Nous avons ensuite utilisé EXP3 au sein de NeuralBandit2 et de NeuralBandit3 afin de sélectionner en ligne le meilleur paramétrage de réseaux. Cette approche est concluante, permettant d'obtenir des courbes de regret très proches de celles des meilleurs modèles et possède l'avantage d'être trivialement parallélisable. Bien qu'offrant des performances similaires à celles de NeuralBandit2 sur le premier jeu de données, NeuralBandit3 obtient un regret légèrement supérieur sur la deuxième expérimentation. Il semblerait en effet que la différentiation des modèles suivant les actions n'apporte pas de gain sur le jeu de données Forest Covert Type. En revanche, nous avons aussi montré empiriquement que NeuralBandit3 converge plus rapidement en cas de concept drift. Ces premiers résultats expérimentaux laissent penser que les réseaux de neurones sont des candidats sérieux à la résolution du problème des bandits contextuels. Ils s'affranchissent cependant de la contrainte de linéarité au détriment de la présence de borne sur le regret, la fonction de coût étant non convexe. Au cours de travaux futurs nous pourrons poser de manière plus formelle le problème de sélection de modèle afin de rentrer dans le cadre des bornes de regret d'EXP3 et nous intéresser aux applications des algorithmes de bandits à la sélection de paramètres continus.

# Références


[ACB98] Peter Auer and Nicolò Cesa-Bianchi. On-line learning with malicious noise and the closure algorithm. *Ann. Math. Artif. Intell.*, 23(1-2) :83–99, 1998.

[ACBF02] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3) :235–256, 2002.

[ACBFS02] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1) :48–77, 2002.

[AG12] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. *CoRR*, 2012.

[BL05] Léon Bottou and Yann LeCun. On-line learning for very large datasets. *Applied Stochastic Models in Business and Industry*, 21(2) :137–151, 2005.

[BMSS08] Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. Online op-

timization in x-armed bandits. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *NIPS*, pages 201–208. Curran Associates, Inc., 2008.

[CKRU08] Deepayan Chakrabarti, Ravi Kumar, Filip Radlinski, and Eli Upfal. Mortal multi-armed bandits. In *NIPS*, pages 273–280, 2008.

[CLRS11] Wei Chu, Lihong Li, Lev Reyzin, and Robert E. Schapire. Contextual bandits with linear payoff functions. In Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudík, editors, *AISTATS*, volume 15 of *JMLR Proceedings*, pages 208–214. JMLR.org, 2011.

[DHK+11] Miroslav Dudík, Daniel Hsu, Satyen Kale, Nikos Karampatziakis, John Langford, Lev Reyzin, and Tong Zhang. Efficient optimal learning for contextual bandits. *CoRR*, abs/1106.2369, 2011.

[FU12] Raphaël Feraud and Tanguy Urvoy. A stochastic bandit algorithm for scratch games. In Steven C. H. Hoi and Wray L. Buntine, editors, *ACML*, volume 25 of *JMLR Proceedings*, pages 129–143. JMLR.org, 2012.

[FU13] Raphaël Feraud and Tanguy Urvoy. Exploration and exploitation of scratch games. *Machine Learning*, 92(2-3) :377–401, 2013.

[GS10] Romaric Gaudel and Michèle Sebag. Feature selection as a one-player game. In Johannes Fürnkranz and Thorsten Joachims, editors, *ICML*, pages 359–366. Omnipress, 2010.

[HSW89] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5) :359–366, July 1989.

[KKM12] Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson Sampling : An Asymptotically Optimal Finite Time Analysis. In *Algorithmic Learning Theory, Proc. of the 23rd International Conference (ALT)*, volume LNCS 7568, pages 199–213, Lyon, France, 2012. Springer.

[KNMS08] Robert D. Kleinberg, Alexandru Niculescu-Mizil, and Yogeshwer Sharma. Regret bounds for sleeping experts and bandits. In *COLT*, pages 425–436, 2008.

[KS06] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *ECML*, volume 4212 of *Lecture Notes in Computer Science*, pages 282–293. Springer, 2006.

[KSST08] Sham M. Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. Efficient bandit algorithms for online multiclass prediction. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 440–447, New York, NY, USA, 2008. ACM.

[LCLS10] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. *CoRR*, 2010.

[LR85] T. L. Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1) :4–22, 1985.

[LZ07] John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *NIPS*. Curran Associates, Inc., 2007.

[MKS+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, 2013.

[RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing : Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[Ros58] Frank Rosenblatt. The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6) :386–408, 1958.

[SAL+11] Yevgeny Seldin, Peter Auer, François Laviolette, John Shawe-Taylor, and Ronald Ortner. Pac-bayesian analysis of contextual bandits. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *NIPS*, pages 1683–1691, 2011.

[Tes02] Gerald Tesauro. Programming backgammon using self-teaching neural nets. *Artificial Intelligence*, 134 :181–199, 2002.

[Tho33] W.R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25 :285–294, 1933.